POST 47- GEN AI

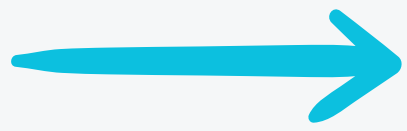
GENERATIVE AI
FOR ALL

Building a Chatbot using OpenAI's API with Python

DINESHLAL

Introduction

- Chatbots are revolutionizing how we interact with technology, from customer service to automating tasks.
- This refined guide will walk you through building a chatbot using OpenAI's powerful API and Python.
- We'll focus on providing clear explanations, practical examples, and helpful tips for students and early career professionals in data science and consulting.



Why Chatbots Matter

- Imagine a consultant who can instantly access vast knowledge bases to answer client questions or a data scientist who can automate data analysis with a simple conversation. That's the power of AI-driven chatbots! They can:
  - Enhance customer engagement
  - **Example**: A retail chatbot can guide customers through product selection based on their preferences.

Why Chatbots Matter

- Imagine a consultant who can instantly access vast knowledge bases to answer client questions or a data scientist who can automate data analysis with a simple conversation. That's the power of AI-driven chatbots! They can:
- **Enhance customer engagement**: Provide 24/7 support, personalized recommendations, and instant responses.
- **Example**: A retail chatbot can guide customers through product selection based on their preferences.

Prerequisites

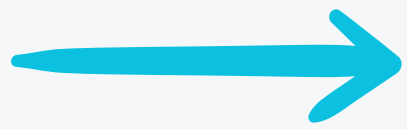
- Before we dive in, make sure you have the following:
- **Python development environment (Python 3.7+):** This is where you'll write and run your chatbot code.
- **OpenAI account and API key:** This gives you access to OpenAI's language models. You can get an API key from the OpenAI platform.
- **Basic understanding of Python and REST APIs:** Familiarity with Python syntax and how APIs work will be helpful.

Step 1: Setting up Your OpenAI API Key

Your API key is like a password that grants access to OpenAI's models. Keep it confidential!

Instructions:

1. Visit the OpenAI platform. - https://platform.openai.com/docs/overview
2. Sign in or create an account.
3. Go to the API section and generate your API key.
4. Store your API key securely, preferably in an environment variable.



Step 1: Setting up Your OpenAI API Key

Code Implementation:

```
import openai

# Set your API key
openai.api_key = "your_api_key_here"  # Replace with your actual API key
```

Step 2: Installing the Necessary Libraries

Code Implementation:

```
import openai
from rich import print
from rich.console import Console

console = Console()
```

Step 3: Creating a Conversational Loop

The core of a chatbot is its ability to listen and respond. We'll achieve this with a conversational loop.

The code in next page creates a simple loop that:

1. Prompts the user for input.
2. Sends the input to OpenAI's API.
3. Displays the response.
4. Continues until the user types "exit" or "quit."

Step 3: Creating a Conversational Loop

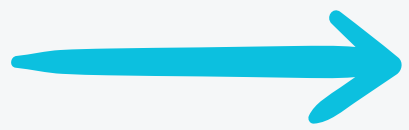
```
import openai
from rich.console import Console

# Initialize console
console = Console()

# Set your API key
openai.api_key = "your_api_key_here"  # Replace with your actual
API key

# Conversational loop
def chatbot_loop():
    console.print("[bold green]Chatbot: Hello! How can I help
you today?[/bold green]")
    while True:
        user_input = console.input("[bold cyan]You: [/bold
cyan]")
        if user_input.lower() in ("exit", "quit"):
            console.print("[bold red]Chatbot: Goodbye![/bold
red]")
            break
        response = openai.Completion.create(
            engine="text-davinci-003",  # You can explore other
engines
            prompt=user_input,
            max_tokens=150  # Adjust the length of the response
        )
        console.print(f"[bold green]Chatbot:
{response['choices'][0]['text'].strip()}[/bold green]")

# Start the chatbot
if __name__ == "__main__":
    chatbot_loop()
```

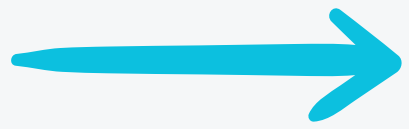



Step 4: Managing Context for Meaningful Conversations

Imagine a human conversation where you forget what was said a few sentences ago. It wouldn't make much sense! Similarly, our chatbot needs to remember the conversation history.

In the improved version shared in next page:

- We use a conversation_history list to store the conversation.
- Each interaction is added to the list as a message with a role ("user" or "assistant").
- The entire history is sent to the API, allowing the model to understand the context.

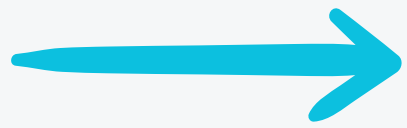


Step 4: Managing Context for Meaningful Conversations

```
import openai
from rich.console import Console

# Initialize console
console = Console()

# Set your API key
openai.api_key = "your_api_key_here"  # Replace with your actual API key
```

Step 4: Managing Context for Meaningful Conversations

```
# Conversational loop with context
def chatbot_loop():
    console.print("[bold green]Chatbot: Hello! How can I help you
today?[/bold green]")
    conversation_history = []

    while True:
        user_input = console.input("[bold cyan]You: [/bold cyan]")
        if user_input.lower() in ("exit", "quit"):
            console.print("[bold red]Chatbot: Goodbye![/bold
red]")
            break

        # Append user input to conversation history
        conversation_history.append({"role": "user", "content":
user_input})

        response = openai.ChatCompletion.create(
            model="gpt-4",  # You can use other models like
gpt-3.5-turbo
            messages=conversation_history
        )

        bot_response = response['choices'][0]['message']
['content']
        console.print(f"[bold green]Chatbot: {bot_response}[/bold
green]")

        # Append bot response to conversation history
        conversation_history.append({"role": "assistant",
"content": bot_response})

# Start the chatbot
if __name__ == "__main__":
    chatbot_loop()
```

Conclusion

- Congratulations! **You've built a basic chatbot with Python and OpenAI's API.** This is a great foundation for further exploration. Here are some ideas to enhance your chatbot:
- **Fine-tuning:** Train the model on specific data to improve its performance in a particular domain (e.g., finance, healthcare).

Conclusion

- **Personalization**: Use user data to tailor responses and create a more engaging experience.
- **Integration**: Connect your chatbot to messaging platforms like Slack or Telegram.
- **Deployment**: Deploy your chatbot as a web service to make it accessible to users.

THANK YOU

- Special thanks to Gemini and ChatGPT for all the help on content
- Follow along for more informative articles in Generative AI space

DINESHLAL